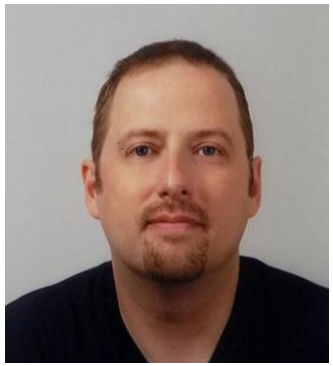
Agile & testing at TomTom Custom Systems

Looking back and going forward

Whoami?

And what does he actually *do*?



- Ronald Hogenboom, born in 1971
- Test Manager at TomTom Custom Systems (Automotive) in EHV
- Proud holder of the ISTQB Practitioner certificate
- Responsible for line Management for test resources, test strategy, test management tool chain and test processes
- Hardware background

Overview

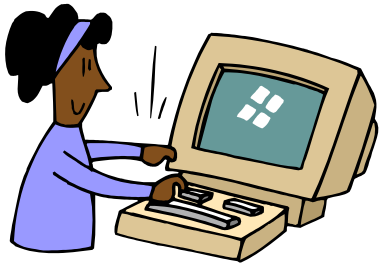
- ‘Classic’ test approach
- Fast Forward
  - We did make some progress you know…
- Introduction of Agile process (scrum)
  - But didn’t we forget something?
- Elements of the TT CS test strategy
- Refinements
- Improvement points

Classic test approach

Party like it's 1999

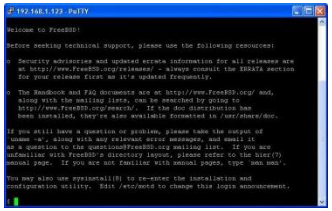
Integrator makes a build

Testers go on test drive



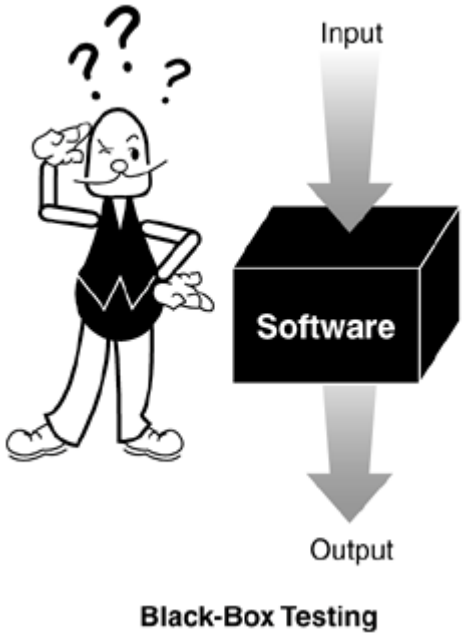
Developers make SW changes

Faxes are written &
problem reports submitted



Classic test approach
Continued

- Purely system based
  - Limited knowledge of architecture
- No requirements
- No test cases
- No clear version control on releases (or anything else)
  - ‘What version are we driving with again?’
    ‘The one Ben made on Tuesday with the positioning fix in it’
    ‘Oh…’



Black-Box Testing

Consequences

- No reproducible test results
- Hard to reproduce problems and difficulty finding root causes
- Very inefficient

Fast forward

- More structured software development
  - Static code checkers
  - Code reviews
  - Proper configuration & release management

- Better qualified testers (no button pushers or failed developers)
  - ISTQB
  - Understand code
  - Can script
  - More white, less black

- Requirements & test management (& tooling) introduced
- Testing earlier in the project lifecycle
- (Automated) testing on more than just system level
- Testing in critical path!

Introduction of scrum

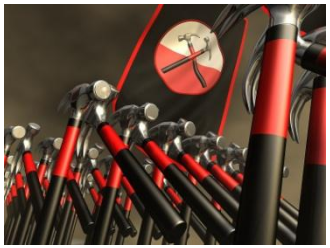
But aren't we forgetting something?

Let's play scrum bingo!

- *Backlog*
- *Self-managing teams*
- *Burn down*
- *Stand-up*
- *Grooming*
- *Retrospective*
- *Shippable release?*

Paradigm shift

'Tear down the wall'



Test and development cannot operate as separate entities

- When 'scrum' was originally introduced, the test team was overlooked. Really.
- However, you cannot have development operate according to scrum and test according to waterfall > How do you handle planning and, more importantly, iterations?
- How will you ever know that you have a shippable release?

Conclusion: Test is also a responsibility of the scrum team!

Elements of the Test Strategy I

It is all about the team…

- Development is done according to the Scrum methodology
- Scrum teams will consist of
  - Team Lead
  - Tech Lead
  - Development engineers
  - *Test Lead*
  - *Test engineers*
- There will be no separate test team / test backlog
- (System) test is the responsibility of the whole team!

Refinement:

- In bigger projects a separate integration / system test scrum team can be defined, but this will still contain Team Lead, Tech Lead, Development engineers *and* Test engineers.

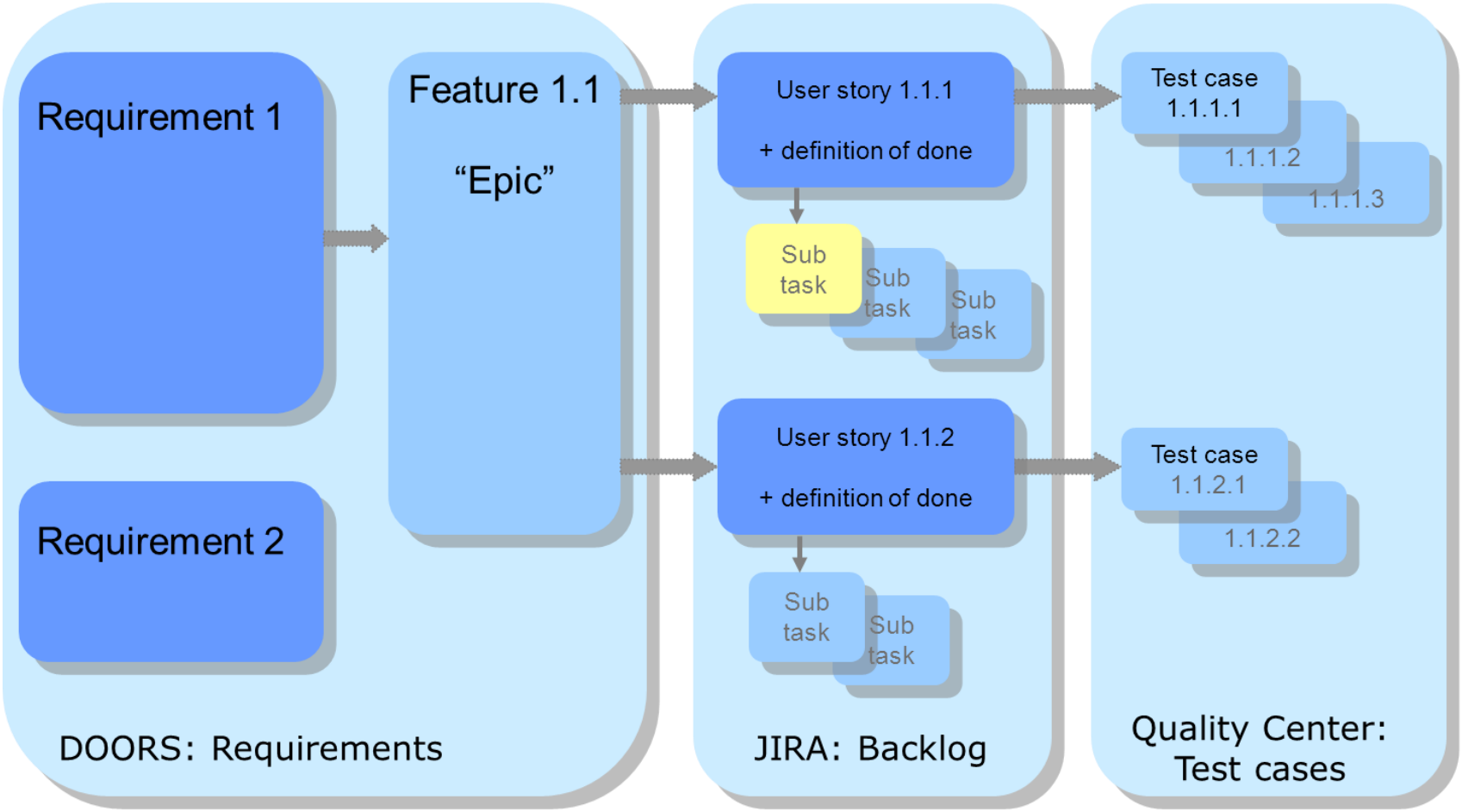
Elements of the Test Strategy II

Test levels at TomTom Custom Systems

Test Level	Test types
System testing *(Low iteration frequency)*	***High risk features:*** • *Manual functional regression tests*
User story testing *(Medium iteration frequency)*	***New functionality:*** • *Initial (new) functional tests* ***Existing functionality:*** • *Automated regression tests, functional and non functional (confidence test)*
Unit testing *(High iteration frequency)*	• *Automated regression tests (smoke test)* ***New functionality*** • *Newly created unit tests* ***Existing functionality:*** • *Existing unit tests*

Elements of the Test Strategy III

Test tool chain & hierarchy of entities at TomTom Custom Systems



Refinements I

Aspects of Kanban

- Do not let your test tasks pile up
- Testing is (also) a team responsibility
- Bottom line: Developers should also pick up testing tasks
- Question: Dies this mean that testers should also pick up developer's tasks?

Room for further improvement

Enough here for another presentation…

- Consistent strategy for unit testing
- Define requirements on more than one level
  - Functional decomposition
- Better measurements
  - Better view on early testing and return on investment
  - Code coverage
  - ‘Slip through’ ratios
- Enforcing the process
- Acting on test results

Thank you

Any questions?